

Some Contributions to Sequential Monte Carlo Methods for Option Pricing

BY DEBORSHEE SEN ¹, AJAY JASRA & YAN ZHOU

Department of Statistics & Applied Probability, National University of Singapore, Singapore, 117546, SG.

Email: deborshee.sen@u.nus.edu; staja@nus.edu.sg; stazhou@nus.edu.sg

Abstract

Pricing options is an important problem in financial engineering. In many scenarios of practical interest, financial option prices associated to an underlying asset reduces to computing an expectation w.r.t. a diffusion process. In general, these expectations cannot be calculated analytically, and one way to approximate these quantities is via the Monte Carlo method; Monte Carlo methods have been used to price options since at least the 1970's. It has been seen in [9, 16] that Sequential Monte Carlo (SMC) methods are a natural tool to apply in this context and can vastly improve over standard Monte Carlo. In this article, in a similar spirit to [9, 16] we show that one can achieve significant gains by using SMC methods by constructing a sequence of artificial target densities over time. In particular, we approximate the optimal importance sampling distribution in the SMC algorithm by using a sequence of weighting functions. This is demonstrated on two examples, barrier options and target accrual redemption notes (TARN's). We also provide a proof of unbiasedness of our SMC estimate.

Key words: Diffusions; Sequential Monte Carlo; Option Pricing

AMS Subject Classification: Primary 91G60; Secondary 65C05.

Acknowledgements

AJ was supported by a Singapore Ministry of Education Academic Research Fund Tier 1 grant (R-155-000-156-112) and is affiliated with the RMI and CQF at NUS. YZ was supported by a Singapore Ministry of Education Academic Research Fund Tier 2 grant (R-155-000-143-112).

1 Introduction

A basic (or *vanilla*) option is a financial product which provides the holder of the option with the right to buy or sell a specified quantity of an underlying asset at a fixed price on or before the expiration date of the option. There are many more complex options (called *exotic* options) in use today; these are often of more practical interest and are harder to deal with. In many scenarios of practical interest and as we shall assume in this article, the value of the underlying asset can be described by a diffusion process; in a complete market, the value of the option can be expressed as the expectation under the risk neutral probability of a functional of the paths of the underlying diffusion process. In general these expectations cannot be calculated analytically. The Monte Carlo (MC) method is a standard approach used to approximate these quantities and it has been extensively used in option pricing since [3]. Subsequently, a wide variety of Monte Carlo approaches have been applied ([14] provides a thorough introduction).

The importance of Monte Carlo for option pricing against other numerical approaches is its ability to deal with high-dimensional integrals. This is either in the time parameter of the option (path-dependent options) or in the dimension of the underlying (basket of options), and more generally in both. However, it has been noted in the option pricing literature that standard Monte Carlo estimates can suffer from high variability. It has been seen in [9, 16, 17] that, in many situations of practical interest, Sequential Monte Carlo (SMC) approaches can vastly improve over more standard Monte Carlo techniques.

Sequential Monte Carlo methods are a general class of methods to sample from a sequence of distributions of increasing dimensions which have extensively been used in engineering, statistics, physics and other

¹Corresponding author.

domains. [11] provides an introduction and shows how essentially all methods for particle filtering can be interpreted as some special instances of a generic SMC algorithm. SMC methods make use of a sequence of proposal densities to sequentially approximate the targets via a collection of M samples, termed particles. In most scenarios it is not possible to use the distribution of interest as a proposal. Therefore, one must correct for the discrepancy between proposal and target via importance weights. In the majority of cases of practical interest, the variance of these importance weights increases with algorithmic time. This can, to some extent, be dealt with a resampling procedure consisted of sampling with replacement from the current weighted samples and resetting them to $1/M$ (adaptive resampling). The variability of the weights is often measured by the effective sample size (ESS). Several convergence results, as M grows, have been proved [4, 5, 6, 12]. SMC methods have also recently been proven to be stable in certain high-dimensional contexts [1].

The main contributions of this paper are as follows. We develop the formal framework of *weighting functions*; this technique has already been used, implicitly or explicitly, in [8, 9, 10, 16]. Exploiting this framework, we develop tailored methods for pricing of barrier options in high dimensional settings. It is also applied to the pricing of Target Accrual Redemption Note (TARN) which are another widely traded kind of path dependent options that are notoriously difficult to accurately value. On the theoretical side, we provide with a proof of the unbiasedness of the SMC estimates when an adaptive resampling scheme is used.

This paper is structured as follows. In Section 2 we provide background details on option pricing. In Section 3 we give a basic summary of SMC methods. In Section 4 we give the weighting functions framework and its application in the context of our option pricing problems. In Section 5 our methods are illustrated numerically. The appendix gives the proof of unbiasedness of our SMC estimate in the adaptive resampling case.

In the remainder of this article, we use the notation \mathbb{R}^d to denote the d -dimensional Euclidean space and $\mathbb{R}_+^d \equiv (0, \infty)^{\otimes d}$. A normal distribution with mean μ and variance σ^2 is denoted by $N(\mu, \sigma^2)$ and its density at x is denoted by $\varphi(x; \mu, \sigma^2)$. I_d denotes the d -dimensional identity matrix. E denotes expectation.

2 Option pricing

Options come in two basic kinds - call and put. Call options give the right to buy and put options give the right to sell. In this context, there are two main kinds of options - American and European. American options can be exercised at any time prior to expiration whereas European options can be exercised only at expiration. We focus on European options in this paper. European call/put options are known as vanilla options since they are relatively simple in structure. An exotic option is an option which has features making it more complex than commonly traded vanilla options. Path-dependent options are an example, in which case the payoff depends on the value of the underlying at some (or all) time points prior to the expiration date. We consider two kinds of path-dependent options in this paper, namely barrier options and TARN's, which we shall describe shortly.

2.1 Model

Consider a collection of d underlying assets; this is also known as a basket. We denote by $\mathbf{R}_t \in \mathbb{R}^d$ the value of the assets in the basket. \mathbf{R}_t is typically modelled by a diffusion process. One such process is a Black-Scholes model with a drift and a volatility

$$d\mathbf{R}_t = \boldsymbol{\mu}(t, \mathbf{R}_t)dt\mathbf{R}_t + \boldsymbol{\sigma}(t, \mathbf{R}_t)\mathbf{R}_td\mathbf{W}_t, \quad (1)$$

where $\boldsymbol{\mu} : \mathbb{R}_+ \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is the drift function, $\boldsymbol{\sigma} : \mathbb{R}_+ \times \mathbb{R}^d \rightarrow \mathbb{R}^d$ is the volatility and \mathbf{W}_t denotes a Brownian motion in \mathbb{R}^d with mean $\mathbf{0}$ and covariance matrix Σ . It is reasonable to assume that \mathbf{W}_t is normalized, that is, $\Sigma_{i,i} = 1$ for all i ; this assumption is valid because the scale factor can be included in the volatility term.

There is an interest rate r , which can depend on time as well.

In general, it is hard to analytically work with (1) except in simple scenarios. This has lead to several discretization methods being available in literature ([14]) with varying levels of accuracy and complexity. One of the most widely used discretization methods is the Euler-Maruyama discretization and we work with it in this paper; however other discretization schemes could also be used which could lead to a lower bias. Consider discretized time points $0 = t_0 < t_1 < \dots < t_N = T$. By letting \mathbf{S} denote the logarithm \mathbf{R} and writing \mathbf{S}_n in place of \mathbf{S}_{t_n} , the Euler-Maruyama discretization of (1) is

$$\mathbf{S}_{n+1} = \mathbf{S}_n + \left(\boldsymbol{\mu}(n, \mathbf{S}_n) - \frac{1}{2} \boldsymbol{\sigma}^2(n, \mathbf{S}_n) \right) \delta t_n + \boldsymbol{\sigma}(n, \mathbf{S}_n) \sqrt{\delta t_n} \mathbf{Z}_n \quad (2)$$

for $n = 1, \dots, N$, where $\mathbf{Z}_n \sim N_d(\mathbf{0}, \Sigma)$, $\mathbf{0}$ denotes the origin in \mathbb{R}^d , and $\delta t_n = t_{n+1} - t_n$ ².

We do not focus on the level of discretization here. Given a particular discretization, we apply our methods to it. The methods developed here could however be used in a multilevel setup (as in [13]); we do not explore this further in this paper.

We assume the drift $\boldsymbol{\mu} = \mathbf{0}$ in order to keep things simple.³ We work with two cases, one where the volatility $\boldsymbol{\sigma}$ is a constant, and another where it depends on the price of the underlying. We now describe two kinds of path-dependent options and we shall later demonstrate our methods on these.

2.2 Barrier options

A barrier option is an exotic derivative, typically an option, on the underlying asset(s) whose value(s) on reaching pre-set barrier levels either springs the option into existence or extinguishes an already existing option. Barrier options exist for baskets as well and the barrier conditions may in general be defined as a function of the underlying assets. For example, the function of the underlying assets could be the mean or could be the maximum of their values. There are two kinds of barrier options:

- when the option springs into existence if a function of the underlying asset values breaches prespecified barriers, it is referred to as being ‘knocked-in’, and
- when the option is extinguished if a function of the underlying asset values breaches prespecified barriers, it is referred to as being ‘knocked-out’.

We consider knocked-out options. These are options that are ‘alive’ as long as the values of the underlying satisfy barrier conditions at some (or all) time points prior to the expiration. If the barrier condition is breached, the option ‘dies’ leading to a zero payoff. If the option is still ‘alive’ at the expiration date, then it gives a payoff that is akin to either a call or a put option (depending on the option type).

We consider a basket of d underlying assets and a barrier option based on these. Barrier options are hard to price using standard MC in the sense that most (if not all) of the particles lead to a zero payoff and this contributes to a high variance of the final estimate. Because of the sequential nature of the evolution of the asset values over time, this is a natural example of a setting where SMC methods can be applied; indeed [9] talks about how SMC methods can be used in this context. We extend their method and show that one can obtain significant gains by choosing the h_n ’s in Section 5 of their paper even by heuristic methods.

²We have a slight abuse of notation in the above, wherein we have used $\boldsymbol{\mu}(n, \mathbf{S}_n)$ and $\boldsymbol{\sigma}(n, \mathbf{S}_n)$ to denote $\boldsymbol{\mu}(t_n, \mathbf{R}_{t_n})$ and $\boldsymbol{\sigma}(t_n, \mathbf{R}_{t_n})$ respectively.

³If $\boldsymbol{\mu}$ is a constant other than $\mathbf{0}$, then it is trivial to extend the methods we propose. If it is a function of the asset value, we could do things similar to what we do in the local volatility model considered later.

2.3 Target Accrual Redemption Note

A Target Accumulation Redemption Note (TARN) provides a capped sum of payments over a period with the possibility of early termination (knockout) determined by target levels imposed on the accumulated amounts. A certain amount of payment is made on a series of cash flow dates (referred to as fixing dates) until the target level is breached. The payoff function of a TARN is path dependent in that the payment on a fixing date depends on the spot value of the asset as well as on the accumulated payment amount up to the fixing date. Typically, commercial software solutions for pricing TARN's are based on the MC method.

There are different versions of TARN products used in FX trading. For simplicity, we consider here a specific form of TARN's. Consider a sequence of fixing dates $0 < T_1 < T_2 < \dots < T_p = T$ and a function $f : \mathbb{R}^d \rightarrow \mathbb{R}$. The function f is decomposed into its positive and negative parts as $f = f^+ - f^-$. Gain and loss processes are defined as follows:

$$G_k = \sum_{i=1}^k f^+(\mathbf{R}_{T_i}) \quad \text{and} \quad L_k = \sum_{i=1}^k f^-(\mathbf{R}_{T_i}),$$

these are the amounts of positive and negative cashflows respectively. There are two cashflow cutoffs Γ_G and Γ_L . Stopping times $\tau^{(L)}$ and $\tau^{(G)}$ are defined as

$$\tau^{(L)} := \inf\{k : L_k \geq \Gamma_L\} \quad \text{and} \quad \tau^{(G)} := \inf\{k : G_k \geq \Gamma_G\},$$

these are the first times when the positive and negative cash flows cross their cut-offs respectively. The overall stopping time τ is defined as

$$\tau := \inf\{\tau^{(L)}, \tau^{(G)}, p\},$$

which is the first time either the positive or the negative cash flows cross a cut-off. The price of the TARN is the expected value of the overall cash flow, $\mathbb{E}[\sum_{i=1}^{\tau} f(\mathbf{R}_{T_i})]$. Here we have assumed that the interest rate is 0, if it wasn't then the expectation would be a weighted sum with the weights corresponding to discounting factors. The reason we assume the interest rate is 0 will be explained in Section 4.3. The main difficulty in applying standard MC methods in this scenario arises from the fact that the function f may be discontinuous. SMC methods can be used instead, which we shall show later.

3 Sequential Monte Carlo methods

SMC methods are a general class of MC methods that sample sequentially from a sequence of target probability densities $\{\pi_n(x_{1:n})\}_{n \geq 1}$ of increasing dimension, where each distribution $\pi_n(x_{1:n})$ is defined on the product space \mathcal{X}^n . Writing

$$\pi_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{Z_n},$$

it is required only that $\gamma_n : \mathcal{X}^n \rightarrow \mathbb{R}$ be known pointwise. In particular, the normalizing constants $Z_n = \int_{\mathcal{X}^n} \gamma_n(x_{1:n}) dx_{1:n}$ may be unknown and an estimate is obtained by the SMC method. SMC provides an approximation of $\pi_1(x_1)$ and an estimate of Z_1 at time 1, then an estimate of $\pi_2(x_{1:2})$ and an estimate of Z_2 at time 2, and so on. These methods work by propagating a collection of M particles using a sequence of importance sampling and resampling steps. In what follows, superscript (m) shall denote the m -th particle. When we write an operation with superscript (m) , we mean that it happens for all M particles. A proposal density $q_n(x_{1:n})$ is selected and particles are proposed from this. The proposal density has the following structure:

$$q_n(x_{1:n}) = q_{n-1}(x_{1:n-1})q_n(x_n|x_{1:n-1}) = q_1(x_1) \prod_{k=1}^n q_k(x_k|x_{1:k-1}).$$

After proposing particles, associated *unnormalized* weights are computed recursively using the decomposition

$$w_n(x_{1:n}) = \frac{\gamma_n(x_{1:n})}{q_n(x_{1:n})} = \frac{\gamma_{n-1}(x_{1:n-1})}{q_{n-1}(x_{1:n-1})} \frac{\gamma_n(x_{1:n})}{\gamma_{n-1}(x_{1:n-1})q_n(x_n|x_{1:n-1})}.$$

These can be written in the form

$$w_n(x_{1:n}) = w_{n-1}(x_{1:n-1})\alpha_n(x_{1:n}) = w_1(x_1) \prod_{k=1}^n \alpha_k(x_{1:k}),$$

where the *incremental weight* function $\alpha_k(x_{1:k})$ is given by

$$\alpha_k(x_{1:k}) = \frac{\gamma_k(x_{1:k})}{\gamma_{k-1}(x_{1:k-1})q_k(x_k|x_{1:k-1})}.$$

These are computed for each particle. The weights $w_n(x_{1:n})$'s are unnormalized because they do not add up to 1. They are normalized by dividing by their sum and the normalized weights are denoted by W_n 's.

Once we obtain a collection of M weighted particles, they are *resampled* according to their weights $\{W_n^{(m)}\}_{m=1}^M$. This is a crucial step which ensures that the system doesn't collapse to very few particles with very high weights. Resampling indices $\{I_n^{(m)}\}_{m=1}^M$ are chosen such that $\mathbb{P}(I_n^{(m)} = j) = W_n^{(j)}$. The system is then updated by setting $\bar{X}_{1:n}^{(m)} \leftarrow X_{1:n}^{(I_n^{(m)})}$ and $\bar{W}_n^{(m)} \leftarrow 1/M$ for $1 \leq m \leq M$. Resampling is expensive and in practice is done only if the variance of the weights is high. One such method is to use the Effective Sample Size (ESS), defined at time n as

$$\text{ESS} = \frac{1}{\sum_{m=1}^M \left(W_n^{(m)}\right)^2}$$

and to perform resampling at time n if the ESS falls below a certain threshold, usually taken to be $M/2$ or $M/3$. Algorithm 1 describes a general SMC algorithm.

We have two approximations of $\pi_n(x_{1:n})$:

$$\begin{aligned} \hat{\pi}_n(x_{1:n}) &= \sum_{m=1}^M W_n^{(m)} \delta_{X_{1:n}^{(m)}}(x_{1:n}) \text{ and} \\ \bar{\pi}_n(x_{1:n}) &= \sum_{m=1}^M \bar{W}_n^{(m)} \delta_{\bar{X}_{1:n}^{(m)}}(x_{1:n}), \end{aligned}$$

which are equal if no resampling is used at time n . Here δ denotes the Dirac delta function. We can also estimate Z_n/Z_{n-1} through

$$\widehat{\frac{Z_n}{Z_{n-1}}} = \sum_{m=1}^M \bar{W}_{n-1}^{(m)} \alpha_n(X_{1:n}^{(m)}).$$

4 Using weighting functions

4.1 In SMC

The goal of using weighting functions is to create a sequence of intermediate target densities that try to approximate the optimal importance sampling density to guide particles towards regions of interest. Consider a discrete time stochastic process $X_{1:N}$ on a general space \mathcal{X}^N . Suppose that we want to estimate the

Algorithm 1 A generic SMC algorithm

- 1: Set initial weights $\bar{W}_0^{(m)} = 1/M$ and initial estimate of normalizing constant $\hat{C}_0 = 1$.
 - 2: **for** $n = 1$ **do**
 - 3: Sample $X_1^{(m)} \sim q_1(x_1)$,
 - 4: compute unnormalized weights $w_1(X_1^{(m)})$,
 - 5: update estimate of normalizing constant $\hat{C}_1 = \hat{C}_0 \times \sum_{m=1}^M w_1(X_1^{(m)})$,
 - 6: compute normalized weights $W_1^{(m)} \propto w_1(X_1^{(m)})$,
 - 7: **if** resampling criterion is satisfied **then**
 - 8: resample $\{W_1^{(m)}, X_1^{(m)}\}_{m=1}^M$ to obtain M equally-weighted particles $\{1/M, \bar{X}_1^{(m)}\}_{m=1}^M$ and set $\{\bar{W}_1^{(m)}, \bar{X}_1^{(m)}\}_{m=1}^M \leftarrow \{1/M, \bar{X}_1^{(m)}\}_{m=1}^M$,
 - 9: **else**
 - 10: set $\{\bar{W}_1^{(m)}, \bar{X}_1^{(m)}\}_{m=1}^M \leftarrow \{W_1^{(m)}, X_1^{(m)}\}_{m=1}^M$.
 - 11: **for** $n \geq 2$ **do**
 - 12: Sample $X_n^{(m)} \sim q_n(x_n | \bar{X}_{1:n-1}^{(m)})$ and set $X_{1:n}^{(m)} \leftarrow (\bar{X}_{1:n-1}^{(m)}, X_n^{(m)})$,
 - 13: compute incremental weights $\alpha_n(X_{1:n}^{(m)})$ and unnormalized weights $w_n(X_{1:n}^{(m)}) = \bar{W}_{n-1}^{(m-1)} \alpha_n(X_{1:n}^{(m)})$,
 - 14: update estimate of normalizing constant $\hat{C}_n = \hat{C}_{n-1} \times \sum_{m=1}^M w_n(X_{1:n}^{(m)})$,
 - 15: compute normalized weights $W_n^{(m)} \propto w_n(X_{1:n}^{(m)})$,
 - 16: **if** resampling criterion is satisfied **then**
 - 17: resample $\{W_n^{(m)}, X_{1:n}^{(m)}\}_{m=1}^M$ to obtain M equally-weighted particles $\{1/M, \bar{X}_{1:n}^{(m)}\}_{m=1}^M$ and set $\{\bar{W}_n^{(m)}, \bar{X}_{1:n}^{(m)}\}_{m=1}^M \leftarrow \{1/M, \bar{X}_{1:n}^{(m)}\}_{m=1}^M$,
 - 18: **else**
 - 18: set $\{\bar{W}_n^{(m)}, \bar{X}_{1:n}^{(m)}\}_{m=1}^M \leftarrow \{W_n^{(m)}, X_{1:n}^{(m)}\}_{m=1}^M$.
-

expectation of a function of this process $H(X_{1:N})$. The basic Monte Carlo method simulates M independent realizations of the process $\{X_{1:N}^{(m)}\}_{m=1}^M$, where $X_{1:N}^{(m)}$ denotes the m -th realization. $E[H(X_{1:N})]$ is estimated by

$$\frac{1}{M} \sum_{m=1}^M H(X_{1:N}^{(m)}).$$

If H is of the form where $H(X_{1:N})$ is zero over most of the space \mathcal{X}^N and non-zero only on a small subset, most of the $H(X_{1:N}^{(m)})$'s would be zero. This subsequently leads to a high variance of the resulting estimate.

More specifically, since \mathcal{X}^N is the sample space, we write

$$\mathcal{S} := \mathcal{X}^N = \mathcal{S}_1 \cup \mathcal{S}_2,$$

where

$$\mathcal{S}_1 = \{x_{1:N} \in \mathcal{X}^N : H(x_{1:N}) = 0\}$$

and

$$\mathcal{S}_2 = \{x_{1:N} \in \mathcal{X}^N : H(x_{1:N}) \neq 0\}.$$

If $\mathbb{P}(X_{1:N} \in \mathcal{S}_1)$ is much larger than $\mathbb{P}(X_{1:N} \in \mathcal{S}_2)$, simulating M independent realizations will lead to most of them lying in \mathcal{S}_1 . Our goal is to use SMC to simulate more particles from \mathcal{S}_2 . In order to do that, we consider a sequence of positive potential functions $G_n : \mathcal{X}^n \rightarrow \mathbb{R}$, $n = 1, 2, \dots, N$, such that $\prod_{n=1}^N G_n(X_{1:n}) = 1$ and write

$$H(X_{1:N}) = \left[\prod_{n=1}^N G_n(X_{1:n}) \right] H(X_{1:N}).$$

The goal is to choose the G_n 's such that they guide the particles towards being in \mathcal{S}_2 through the weighting and resampling steps of SMC.

When this is done on a path space, the resulting algorithm is sometimes known as tempering. [7] considers this and shows how one can construct an artificial sequence of intermediate target densities on the path space which guide particles towards regions of interest. Doing it on the path space however makes it computationally expensive and we do not work on the path space.

4.2 Barrier options

We consider a discretely monitored barrier option monitored on a series of monitoring dates $T_1 < \dots < T_p = T$ and a sequence of lower and upper barriers $\mathbf{L}_{T_1}, \dots, \mathbf{L}_{T_p} \in \mathbb{R}^d$ and $\mathbf{U}_{T_1}, \dots, \mathbf{U}_{T_p} \in \mathbb{R}^d$ respectively. We suppose that the barrier conditions are that all the underlying asset values lie inside their respective barriers at the monitoring times. This is a simplistic assumption and makes it easier for us to demonstrate our methods; more complicated barrier conditions could also be used.

For ease of notation, we remove the T_i from the barriers and replace it simply by i . Let $X_n = (\mathbf{S}_{n-1}, \mathbf{S}_n) \in \mathbb{R}^{2d}$ for $n \geq 1$ and let H denote the payoff function at time T . The sequence of random variables $X_{1:N}$ then forms a Markov Chain. The price of the barrier option is

$$Q_D = E \left[H(\mathbf{S}_T) \prod_{i=1}^p \mathbf{1} \{ \mathbf{S}_{T_i} \in (\mathbf{L}_i, \mathbf{U}_i) \} \right]^4,$$

⁴We have assumed here that the interest rate is 0. If the interest rate was r , then there would be a factor of $e^{\int_0^T r(t) dt}$ multiplied with Q_D . This is a constant and affects the variance of the estimate only upto a (known) scale factor.

where

$$\mathbf{1}\{\mathbf{S}_{T_i} \in (\mathbf{L}_i, \mathbf{U}_i)\} = \prod_{j=1}^d \mathbf{1}\{S_{T_i,j} \in (L_{i,j}, U_{i,j})\}.$$

$S_{T_i,j}$, $L_{i,j}$ and $U_{i,j}$ denote the j -th components of \mathbf{S}_{T_i} , \mathbf{L}_i and \mathbf{U}_i respectively. In this case,

$$\begin{aligned} \mathcal{S} &= \mathbb{R}^{dN}, \\ \mathcal{S}_2 &= \{\mathbf{s} \in \mathbb{R}^{dN} : s_{T_i,j} \in (L_{i,j}, U_{i,j}), i = 1, 2, \dots, p, j = 1, 2, \dots, d\}, \\ \mathcal{S}_1 &= \mathcal{S} \setminus \mathcal{S}_2. \end{aligned}$$

The authors in [9] introduce an SMC algorithm to estimate the price. The proposal density q is chosen to be density with respect to the underlying discretization. For each time interval (T_{i-1}, T_i) they simulate forward till T_i and then resample particles that breach the barrier condition at time T_i from among the particles that are still inside the barrier. Their algorithm is Algorithm 2. It is commented that they do not use an adaptive version of resampling and instead always resample.

Algorithm 2 SMC For barrier options

- 1: Set initial weights $\bar{W}_0^{(m)} = 1/M$ and initial estimate of normalizing constant $\hat{C}_0 = 1$.
 - 2: **for** $n = 1$ **do**
 - 3: Sample $X_1^{(m)} \sim q_1(x_1)$.
 - 4: **for** $n \geq 2$ **do**
 - 5: Sample $X_n^{(m)} \sim q_n(x_n | \bar{X}_{n-1}^{(m)})$ and set $X_{1:n}^{(m)} \leftarrow (\bar{X}_{1:n-1}^{(m)}, X_n^{(m)})$.
 - 6: **if** $n \in \{T_1, \dots, T_p\}$, say $n = T_k$ **then**
 - 7: Compute unnormalized weights $w_k^{(m)} = W_{k-1}^{(m)} \times \mathbf{1}\{\mathbf{S}_{T_k}^{(m)} \in (\mathbf{L}_k, \mathbf{U}_k)\}$,
 - 8: update estimate of normalizing constant $\hat{C}_k = \hat{C}_{k-1} \times \sum_{m=1}^M w_k^{(m)}$,
 - 9: compute normalized weights $W_k^{(m)} \propto w_k^{(m)}$,
 - 10: resample $\{W_n^{(m)}, X_{1:n}^{(m)}\}_{m=1}^M$ to obtain M equally-weighted particles $\{1/M, \bar{X}_{1:n}^{(m)}\}_{m=1}^M$ and set $\{\bar{W}_n^{(m)}, \bar{X}_{1:n}^{(m)}\}_{m=1}^M \leftarrow \{1/M, \bar{X}_{1:n}^{(m)}\}_{m=1}^M$.
-

The estimated price is

$$\hat{Q}_D = \hat{C}_p \times \sum_{m=1}^M \bar{W}_p^{(m)} H(\bar{\mathbf{S}}_N^{(m)}).$$

Moreover, since essentially only the normalizing constant is being computed, there is no issue of path degeneracy⁵. Resampling paths outside the barriers at the monitoring times from paths which are inside the barriers improves the efficiency of the estimator with respect to the standard MC estimator. It is remarked that for constant volatility, in a Black-Scholes context, one can sample the price to ensure, in one step, that the process always survives; see [15].

If, however, very few particles satisfy the barrier condition at time T_i , then this estimate will also have a high variance. This can happen for example if: the dimension d is high; the barrier condition is a narrow one, i.e. \mathbf{L}_i and \mathbf{U}_i are close to each other; the volatility σ is high; the time intervals (T_{i-1}, T_i) are large. Our goal is to introduce a sequence of positive weighting functions so that they resolve this issue. In order to do so, we write

⁵Path degeneracy is when repeated resampling steps lead to many multiple copies of the same particle $X_{1:N}$. This causes estimates based on the entire paths being unreliable.

$$\begin{aligned}
Q_D &= \mathbb{E} \left[h_0(\mathbf{S}_0) \frac{h_1(\mathbf{S}_1)}{h_0(\mathbf{S}_0)} \times \cdots \times \frac{h_{T_1}(\mathbf{S}_{T_1})}{h_{T_1-1}(\mathbf{S}_{T_1-1})} \times h_{T_1+1}(\mathbf{S}_{T_1+1}) \times \frac{h_{T_1+2}(\mathbf{S}_{T_1+2})}{h_{T_1+1}(\mathbf{S}_{T_1+1})} \times \cdots \right. \\
&\quad \left. \times \frac{h_{T_{p-1}}(\mathbf{S}_{T_{p-1}})}{h_{T_{p-1}-1}(\mathbf{S}_{T_{p-1}-1})} \times h_{T_{p-1}+1}(\mathbf{S}_{T_{p-1}+1}) \times \frac{h_{T_{p-1}+2}(\mathbf{S}_{T_{p-1}+2})}{h_{T_{p-1}+1}(\mathbf{S}_{T_{p-1}+1})} \times \cdots \times \frac{h_{T_p}(\mathbf{S}_{T_p})}{h_{T_p-1}(\mathbf{S}_{T_p-1})} \right] \\
&= h_0(\mathbf{S}_0) \times \mathbb{E} \left[\prod_{n=1}^N G_n(X_n) \right], \tag{3}
\end{aligned}$$

where

$$\begin{aligned}
G_n(X_n) &= G_n(\mathbf{S}_{n-1}, \mathbf{S}_n) = \frac{h_n(\mathbf{S}_n)}{h_{n-1}(\mathbf{S}_{n-1})}, \quad n \notin \{T_1 + 1, T_2 + 1, \dots, T_{p-1} + 1\}, \\
G_{T_i+1}(X_{T_i+1}) &= G_{T_i+1}(\mathbf{S}_{T_i}, \mathbf{S}_{T_i+1}) = h_{T_i+1}(\mathbf{S}_{T_i+1}) \quad \text{for } i = 1, 2, \dots, p-1, \text{ and} \\
h_{T_i}(\mathbf{S}_{T_i}) &= \mathbf{1}\{\mathbf{S}_{T_i} \in (\mathbf{L}_i, \mathbf{U}_i)\} \quad \text{for } i = 1, 2, \dots, p.
\end{aligned}$$

The h_n 's, $n \notin \{T_1, T_2, \dots, T_p\}$, are the sequence of positive weighting functions. In Algorithm 2, they were simply 1. We attempt to choose them more carefully in order to approximate the optimal importance sampling densities. The algorithm is Algorithm 3.

The estimated price is

$$\hat{Q}_D = h_0(\mathbf{S}_0) \times \hat{C}_N \sum_{m=1}^M \bar{W}_N^{(m)} H(\bar{\mathbf{S}}_N^{(m)}). \tag{4}$$

Path degeneracy is again not an issue because we are still essentially estimating the normalizing constant. This estimate is unbiased and a proof is provided in the appendix.

Since h_{T_i} is the indicator of \mathbf{S}_{T_i} being inside the barriers at time T_i , paths which are outside the barriers at the monitoring times are discarded with probability 1 in this case as well. However unlike in Algorithm 2, here we seek to give higher weights to particles which we think have a higher chance of being inside the barriers at the monitoring times. What is being sought while using the weighting functions is an approximation to the optimal importance sampling density, that is, the density of \mathbf{S}_t (at time t) conditional on it surviving at times T_i , $i = 1, 2, \dots, p$. In the case of the barrier options being considered, this corresponds to $\mathbf{S}_{T_i} \in (\mathbf{L}_i, \mathbf{U}_i)$. This is achieved by giving higher weights to particles which have a higher chance of surviving at times T_i . For example, particles which are far away from the barriers have a lower chance of survival than particles which are closer to the barriers. This is the intuitive idea behind our choice of weighting functions, and this is illustrated in Section 5.1.

4.3 TARNs

We consider a TARN based on a single underlying asset. Since the main problem arises when the function f is discontinuous, we consider a discontinuous f to illustrate our methods. Let

$$f(r) = \begin{cases} 2(r - 110) + 20 & \text{if } r > 110 \\ 2(80 - r) + 20 & \text{if } r < 90 \\ -20 & \text{if } 90 \leq r \leq 110 \end{cases}$$

This function has two big jumps at 90 and 110. The negative and positive cashflow cutoffs are $\Gamma_L = 100$ and $\Gamma_G = 200$. We recall that we had earlier assumed the interest rate is 0. This is now justified. The main reason why standard MC cannot be used efficiently in this scenario is as follows. Using MC, most of the particles stay inside (90, 110) for the first five fixing dates. This leads to the contribution of the particle in the MC estimate being -100 . However, an occasional particle escapes (90, 110) within the first five fixing

Algorithm 3 SMC for option pricing with weighting functions

1: Set initial weights $\bar{W}_0^{(m)} = 1/M$ and initial estimate of normalizing constant $\hat{C}_0 = 1$.

2: **for** $n = 1$ **do**

3: Sample $X_1^{(m)} \sim q_1(x_1)$,

4: compute unnormalized weights

$$w_1^{(m)} = W_0^{(m)} \times G_1 \left(X_1^{(m)} \right),$$

5: update estimate of normalizing constant $\hat{C}_1 = \hat{C}_0 \times \sum_{m=1}^M w_1^{(m)}$,

6: compute normalized weights $W_1^{(m)} \propto w_1^{(m)}$,

7: **if** resampling criterion is satisfied **then**

8: resample $\left\{ W_1^{(m)}, X_1^{(m)} \right\}_{m=1}^M$ to obtain M equally-weighted particles $\left\{ 1/M, \bar{X}_1^{(m)} \right\}_{m=1}^M$ and set

$$\left\{ \bar{W}_1^{(m)}, \bar{X}_1^{(m)} \right\}_{m=1}^M \leftarrow \left\{ 1/M, \bar{X}_1^{(m)} \right\}_{m=1}^M,$$

9: **else**

$$\text{set } \left\{ \bar{W}_1^{(m)}, \bar{X}_1^{(m)} \right\}_{m=1}^M \leftarrow \left\{ W_1^{(m)}, X_1^{(m)} \right\}_{m=1}^M.$$

10: **for** $n \geq 2$ **do**

11: Sample $X_n^{(m)} \sim q_n \left(x_n | \bar{X}_{n-1}^{(m)} \right)$ and set $X_{1:n}^{(m)} \leftarrow \left(\bar{X}_{1:n-1}^{(m)}, X_n^{(m)} \right)$,

12: compute unnormalized weights

$$w_n^{(m)} = W_{n-1}^{(m)} \times G_n \left(\mathbf{X}_n^{(m)} \right),$$

13: update estimate of normalizing constant $\hat{C}_n = \hat{C}_{n-1} \times \sum_{m=1}^M w_n^{(m)}$,

14: compute normalized weights $W_n^{(m)} \propto w_n^{(m)}$,

15: **if** resampling criterion is satisfied **then**

16: resample $\left\{ W_n^{(m)}, X_{1:n}^{(m)} \right\}_{m=1}^M$ to obtain M equally-weighted particles $\left\{ 1/M, \bar{X}_{1:n}^{(m)} \right\}_{m=1}^M$ and set

$$\left\{ \bar{W}_n^{(m)}, \bar{X}_{1:n}^{(m)} \right\}_{m=1}^M \leftarrow \left\{ 1/M, \bar{X}_{1:n}^{(m)} \right\}_{m=1}^M,$$

17: **else**

$$\text{set } \left\{ \bar{W}_n^{(m)}, \bar{X}_{1:n}^{(m)} \right\}_{m=1}^M \leftarrow \left\{ W_n^{(m)}, X_{1:n}^{(m)} \right\}_{m=1}^M.$$

dates and contributes a value that is significantly different from -100 because of the big jumps in f . This causes the variance of the MC estimate to be high. Even if the interest rate was positive, this difficulty would still remain. Therefore for simplicity we assume the interest rate to be 0.

In order to go back to the previous notation, let $S = \log(R) \in \mathbb{R}$ and define $g(s) = f(e^s)$. Then let

$$g(S_{1:N}) := 100 + \sum_{i=1}^{\tau} g(S_{T_i})$$

be the new payoff function, where we have written $S_{1:N}$ in place of $(S_{t_1}, S_{t_2}, \dots, S_{t_N})$. By defining g in this way, the problem has been transformed into the format that was being using before. In this case,

$$\begin{aligned} \mathcal{S} &= \mathbb{R}^N, \\ \mathcal{S}_1 &= \left\{ s_{1:N} \in \mathbb{R}^N : L_5 := \sum_{i=1}^5 g^-(s_{T_i}) = 100 \right\}, \\ \mathcal{S}_2 &= \left\{ s_{1:N} \in \mathbb{R}^N : L_5 := \sum_{i=1}^5 g^-(s_{T_i}) < 100 \right\}. \end{aligned}$$

A particle lies in \mathcal{S}_1 if (and only if) it stays within $(90, 110)$ for the first five fixing dates. $\mathbb{P}(\mathcal{S}_1) \gg \mathbb{P}(\mathcal{S}_2)$ if, for example, the volatility is low or the time intervals $T_i - T_{i-1}$ are small. It is noted that this is the opposite of the barrier option case (in which we considered the time intervals and volatility to be large). We again consider a sequence of positive weighting functions h_1, h_2, \dots, h_{T_5} and write

$$g(s_{1:N}) = \left[\prod_{n=1}^{T_5} \frac{h_n(s_n)}{h_{n-1}(s_{n-1})} \right] \times \frac{g(s_{1:N})}{h_{T_5}(s_{T_5})},$$

where $h_0(s_0) \equiv 1$; we are basically doing the same thing as in the barrier option case. The goal is again to guide particles towards being in \mathcal{S}_2 , and we show that this can be achieved through the usage of some simple weighting functions in Algorithm 3.

5 Numerical results

In this section we demonstrate numerically the benefit of using weighting functions. In order to compare the standard deviations of Algorithms 2 and 3, we run them 100 times with 100000 particles in each run. We then look at the standard deviations of the 100 estimates and report the relative standard deviations (the ratio of the standard deviations).

5.1 Barrier options

We consider a barrier option whose asset values evolve independently of each other. This translates to $\Sigma = I_d$. The option type is call. We assume that we can simulate forward one day at a time. A common monitoring strategy is to monitor the underlying assets after every k days for a total of p time periods. In that case, $T_i = ik$ for $i = 1, \dots, p$ and $N = pk$. We choose $k = 540$ and $p = 1$. Algorithm 2 resamples at the end of p days and so resets the system of particles by choosing all resampled particles being inside the barriers. Therefore the gain that we expect by using Algorithm 3 can only be before the system is reset and this is why we choose $p = 1$. In what follows, we refer to Algorithm 2 as ‘MC’ and Algorithm 3 as ‘SMC’.

The algorithms are run on different values of the dimension d and the volatility $\sigma = (\sigma_1, \dots, \sigma_d)$. Recalling (3), the targeted density at any time n is proportional to $h_n(\mathbf{s}_n)p_n(\mathbf{s}_{1:n})$, where $p_n(\mathbf{s}_{1:n})$ denotes the density of $\mathbf{s}_{1:n}$. This implies that the targeted marginal density at time n is proportional to $h_n(\mathbf{s}_n)p_n(\mathbf{s}_n)$, where $p_n(\mathbf{s}_n)$ denotes the marginal density of $\mathbf{s}_{1:N}$ at time n . We denote the targeted density at time n by $\tilde{p}_n(\mathbf{s}_n)$.

5.1.1 Constant volatility model

Since our basket consists of d independent assets, we choose the marginal targeted densities (at different time points) to be the product of d (unnormalized) densities. That is, $\tilde{p}_n : \mathbb{R}^d \rightarrow \mathbb{R}$ is such that $\tilde{p}_n(\mathbf{s}_n) = \prod_{j=1}^d \tilde{p}_{n,j}(s_{n,j})$, where $\mathbf{s}_n = (s_{n,1}, \dots, s_{n,d})$. In the constant volatility case, the marginal density of $\mathbf{S}_{1:N}$ at time n is known and is denoted by $p_n(\mathbf{s}_n) = \prod_{j=1}^d p_{n,j}(s_{n,j})$; this is simply a product of d Gaussians. Therefore the weighting functions are $h_n(\mathbf{s}_n) = \prod_{j=1}^d [\tilde{p}_{n,j}(s_{n,j})/p_{n,j}(s_{n,k})]$.

We choose $\sigma_1 = \dots = \sigma_d = \sigma$. Since our goal is to push particles towards being inside the barriers at time k , we consider a (one dimensional) Brownian bridge with volatility σ tied down at $B_{0,j} = S_{0,j}$ and $B_{k,j} = K_j := (L_j + U_j)/2$. However, since the variance of the Brownian bridge goes to 0 as the time approaches k , we cannot directly use it as it would lead to a high variance. So we add 0.2σ to the standard deviation of the Brownian bridge. It is remarked that 0.2σ is somewhat an arbitrary choice. This is not necessarily the best choice, and we do not claim so; this choice just serves to illustrate the benefit of using weighting functions. We have noticed gains even while using other values, all we need to do is to ensure that the targeted densities don't tend towards being degenerate. In practice, we introduce the weighting functions only after time $n = 2k/3$ because we want to let particles initially explore the space and then give higher weights to particles which are more likely to be inside the barriers at time k .

The results are in Figure 1. We observe a gain in the standard deviation by using these functions. In general, if the volatilities, initial prices and strike prices were different for the different chains, we could have considered a product of d dissimilar Brownian bridge densities as the weighting functions.

Since the actual goal is to estimate the optimal importance density, we try to do it in our simple setup to see how much better it does. The marginal densities of a particle given that it survives at time k can be approximated by Gaussian densities. In fact, this is what we were doing earlier when we choose the targeted distribution to be a Brownian bridge. Instead of working with a Brownian bridge, we can try to estimate the means and the variances of the Gaussian densities by first simulating particles in one dimension and then looking at the means and variance of those that survive.

In this regard, we simulate $M_1 = 10000$ particles in one dimension with volatility $\sigma = 0.08$. We then look at the particles that survive at the end. Given the particles that survive, we look at their marginal means and variances. This approximates the marginal densities (in one dimension) of a particle given that it survives. Since we are in one dimension, the number of particles surviving is much higher than what we would have in higher dimensions (about 3900 out of 10000 in our simulations). This means that the probability of survival in one dimension is about 0.39 for $\sigma = 0.08$, which for example in 10 dimensions would mean a probability of survival of 8.1×10^{-5} (since the chains are independent). We call this the 'optimal' target.

The results are in Figure 2. We can observe an improvement against the standard MC method, relative to Figure 1. However, the gains are not very drastic and serve to illustrate that even the simple intuitive use of a Brownian bridge does well. It is worth noting that even by choosing the volatility to be 0.08 and running the chain in one dimension, we observe a gain for other values of the volatility as well. This suggests that even if the volatilities were different for the different chains, we could have simply run a chain in one dimension with a value of the volatility that is in the range of the volatilities and still get decent results.

A note on running times: we have observed that the running time for MC is less than three times that of SMC (even without having optimized the codes too much). Thus if we were to take running times into account, we could have run MC with $3M$ particles in the same time in which we run SMC with M particles. The standard deviation for MC using $3M$ particles would be $\sqrt{3}$ times less than that of MC with M particles. So even by taking running times into account, SMC outperforms MC. In the last example, in addition we run MC with 10000 particles. We should take this time into account as well, but we run it for only one value

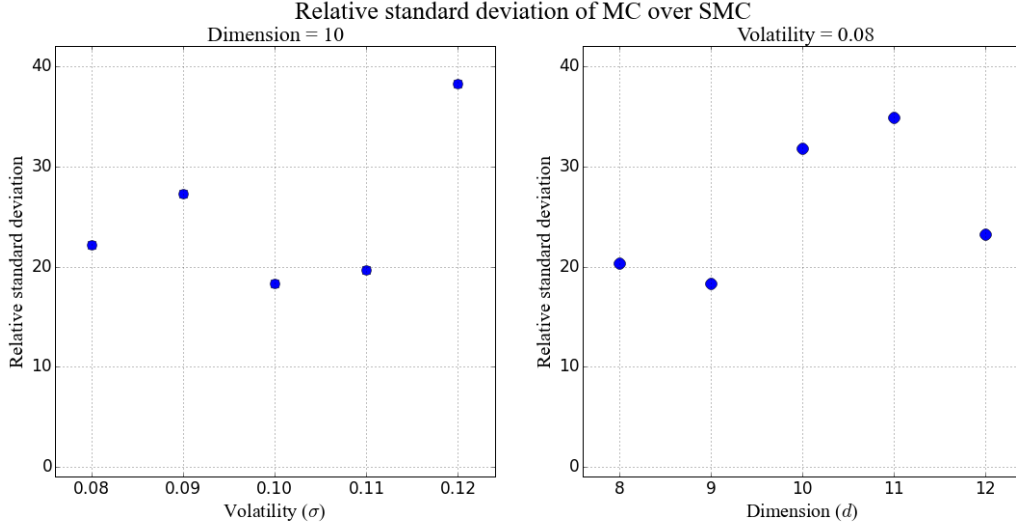


Figure 1: Relative standard deviations of MC over SMC (Algorithm 3) for a barrier option with $m = 1$ and $k = 540$ for a constant volatility model when we target a Brownian bridge; in the left figure, the dimension $d = 10$; in the right figure, the volatility $\sigma = 0.08$.

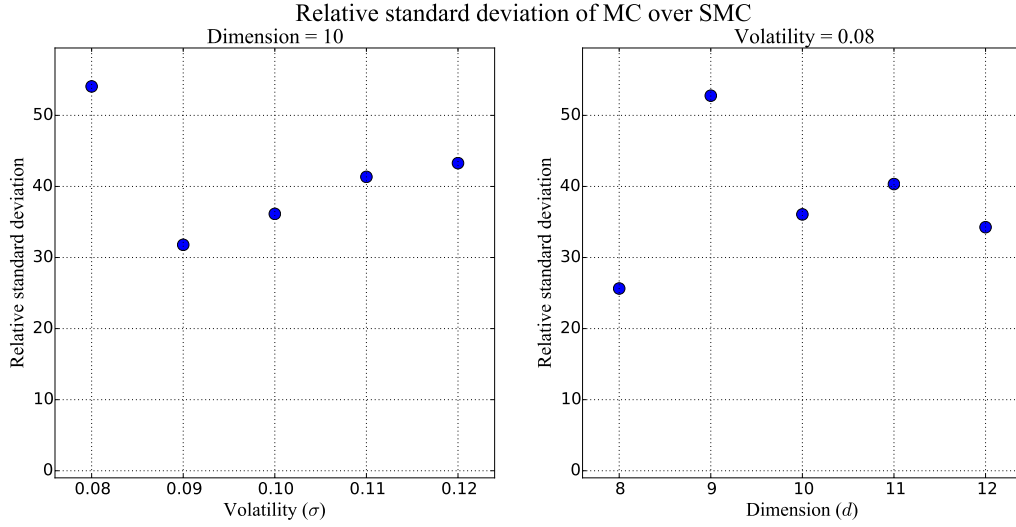


Figure 2: Relative standard deviations of MC over SMC (Algorithm 3) for a barrier option with $m = 1$ and $k = 540$ for a constant volatility model when we target the 'optimal' target; in the left figure, the dimension $d = 10$; in the right figure, the volatility $\sigma = 0.08$.

of the volatility and it is fast. The reason we do not consider the running times in our figures is that we believe it might be possible to optimize the codes even further. The intention is to demonstrate the benefit of using weighting functions in SMC.

5.1.2 Local volatility model

Local volatility models are in practice used more frequently as they are typically more accurate. The local volatility function considered here is a linear interpolation between the values of the volatility σ at 0.12, 0.11, 0.105, 0.101, 0.097, 0.093, 0.098, 0.10, 0.105, 0.11, 0.17 and the values of the underlying R at 10^{-6} , 60, 70, 80, 90, 100, 110, 120, 130, 140, 10^6 . In this case, we do not know the marginal densities of $\mathbf{S}_{1:N}$. For simplicity, we still assume that the volatility functions are the same for all the chains and approximate it in the following way. Consider a single chain. If we knew $E S_1, E S_2, \dots$, then we could approximate $\sigma^2(S_n)$ by $\sigma^2(E S_n)$. We approximate $E S_n$ iteratively as follows (assuming that the time step of the discretization δt_n is constant):

$$\begin{aligned} E[S_n | S_{n-1}] &= S_{n-1} - \frac{1}{2}\sigma^2(S_{n-1})\delta t \\ \Rightarrow E S_n &= E S_{n-1} - \frac{1}{2}\delta t E[\sigma^2(S_{n-1})] \\ &\approx E S_{n-1} - \frac{1}{2}\delta t \sigma^2(E S_{n-1}); \end{aligned} \tag{5}$$

therefore we approximate $E S_n$ and $\sigma^2(E S_n)$ iteratively. We first approximate $E S_1$ by (5); given this estimate, we approximate $E S_2$ by (5) and keep doing this. We approximate $p_n(\mathbf{S}_n)$ by

$$\hat{p}_n(\mathbf{S}_n) = \prod_{j=1}^d \varphi\left(s_{n,j} \middle| E S_{n-1} - \frac{1}{2}\sigma^2(E S_{n-1})\delta t, \sigma^2(E S_{n-1})n\delta t\right)$$

iteratively.

1. We begin with a Brownian bridge weighting function using the previously estimated values of $\sigma(E S_n)$ in the target density and add $0.2\sigma(E S_n)$ to the standard deviation as before.
2. As in the constant volatility model, we simulate $M_1 = 10000$ particles (for the local volatility model) in one dimension and look at the marginal means and variance of the particles that survive. We call this the ‘optimal’ target and target it in the SMC algorithm.

The results are in figure 3. We observe a significant gain, which as expected, gets more significant as the dimension increases. This is because the probability of a particle surviving gets smaller as the dimension increases. The gains are more significant when we target the ‘optimal’. As far as running times are concerned, the observations are the same as in the case of the constant volatility model. Thus Algorithm 3 outperforms Algorithm 2 even upon taking running times into consideration.

5.2 TARNs

We again assume that we can simulate forward one day at a time. We take the number of fixing dates $p = 24$ and the interval between fixing dates $k = 30$ days. This roughly corresponds to a TARN where there is a cash flow at the end of every month for a maximum of two years. We consider different values of the volatility. A particle leads to a zero payoff if it stays within $(-90, 110)$ for the first five months and to a positive payoff if it escapes by the end of the fourth month. We again compare standard MC and Algorithm 3 (referring to this as ‘SMC’). We note that contrary to barrier options, in this case we expect SMC to do better than usual MC for *low* values of the volatility. This is because in this case the region \mathcal{S}_2 has low probability for lower values of the volatility unlike in the barrier option case where \mathcal{S}_2 had low probability for higher values of the volatility. This is why we restrict ourselves to lower values of the volatility in this section than in Section 5.1.

5.2.1 Constant volatility model

When the underlying follows Black-Scholes dynamics with a constant volatility σ , we can directly simulate forward k days at a time using the Euler-Maruyama discretization. We run the algorithms different values of the volatility and report the results. We try three (increasingly more sophisticated, yet intuitive) weighting functions to try and get particles to escape.

In the most naive version, we choose $h_n(s_{1:n})$ to simply be $(s_n - S_0)^2$. This just gives higher weights to particles that are further away from S_0 at the end of the n -th month. We obtain the results in Figure 4.

We then notice that the targeted density at time n is proportional $h_n(s_{1:n})p(s_{1:n})$ for $1 \leq n \leq 5$. If we choose $h_n(s_{1:n})$ to be of the form $h_n(s_n)$, then the marginal of the targeted density at time n is proportional to $h_n(s_n)p_n(s_n)$, where $p_n(s_n)$ is the marginal density of $S_{1:N}$ at time n . Since we target $(s_n - S_0)^2$, we choose $h_n(s_n) = (s_n - S_0)^2/p_n(s_n)$ and this leads to the results in Figure 5. It is seen that SMC with weighting functions does much better in most cases, except in the last few cases. The last few cases correspond to higher volatilities, in which case the probability of a particle escaping increases. SMC does better when the probability of a particle escaping is low, which is what we wanted.

Looking into the problem further, we observe that particles which escape can do so in one of two ways: either to the left or to the right. We fix a value of $\sigma = 0.05$ and run 10000 copies of the chain. Our goal is to look at the exact marginal distribution of particles of particles which have escaped. We observe that approximately 20% of particles escape to the left and 80% escape to the right. This is of course for a fixed σ , but we use these marginal approximations as our target (we use a mixture of normals) and run the SMC algorithm for different values of σ . The results are in Figure 6. Significant gains are observed in this case as well, and are more than the gains before. In this case as well, we are trying to approximate the optimal importance density.

In all of the above three examples in this section, we have observed that the running time for SMC is less than twice that of MC. We can thus have similar conclusions as in the case of barrier options, which shows that SMC outperforms MC even upon taking running times into consideration.

5.2.2 Local volatility model

In this case, the volatility σ is a function of the price of the underlying R_t . We cannot simulate forward k days at a time any more and we simulate forward one day at a time now. In this case the weighting functions are introduced each day for the first 5 time periods.

We consider a volatility function which is minimum at 100 and increases on either side of 100. The value at 100 is 0.036. The values in the (90, 110) are less than 0.04 and are higher outside it. We choose this because a low value of σ would mean less particles escaping, but for the ones that do escape we let explore the space more freely. The local volatility function is a linear interpolation between the values of the volatility σ at 0.055, 0.051, 0.045, 0.041, 0.037, 0.035, 0.038, 0.04, 0.045, 0.05, 0.055 and the values of the underlying R at 10^{-6} , 60, 90, 93, 98, 100, 103, 107, 110, 140, 10^6 .

We start off with the simplest weighting function $h_n(s_{1:n}) = (s_n - S_0)^2$. In this case, we observe that Algorithm 3 has a standard deviation which is 1.83 times lower than that of MC.

Since the marginal density of S_n is unknown, we choose a crude approximation of the marginal and use it. We choose

$$\hat{p}_n(s_n) = \varphi \left(s_n \middle| S_0 + \left(\mu - \frac{1}{2} 0.04^2 \right) \delta t_n, 0.04^2 \delta t_0 \right)$$

to be the approximation. This is the density of s_n if the volatility had been a constant 0.04. We run 10000 copies of the chain and look at the ones which escape towards the left and towards the right. We then choose

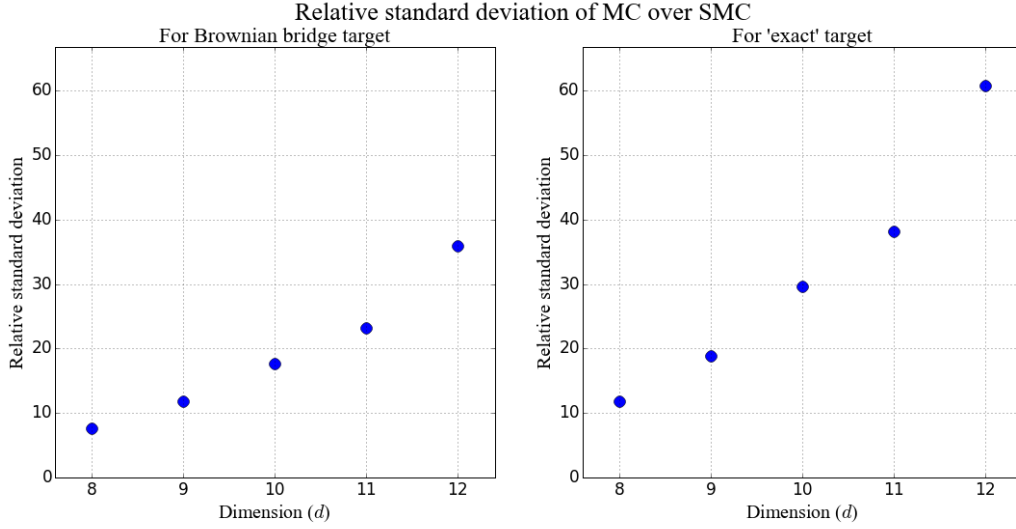


Figure 3: Relative standard deviations of MC over SMC (Algorithm 3) for a barrier option with $m = 1$ and $k = 540$ for local volatility model; the left figure is when we target a Brownian bridge; the right figure is when we target the ‘optimal’.

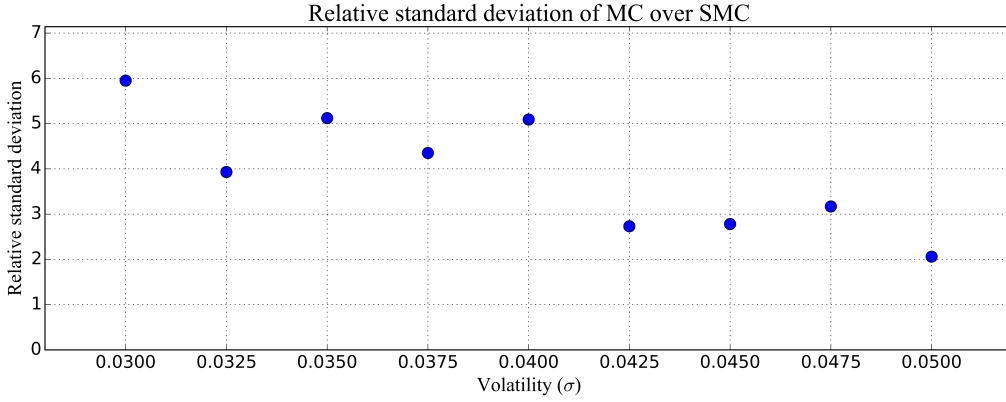


Figure 4: Relative standard deviations of MC over SMC (Algorithm 3) using most naive weighting functions for TARN in the constant volatility case.

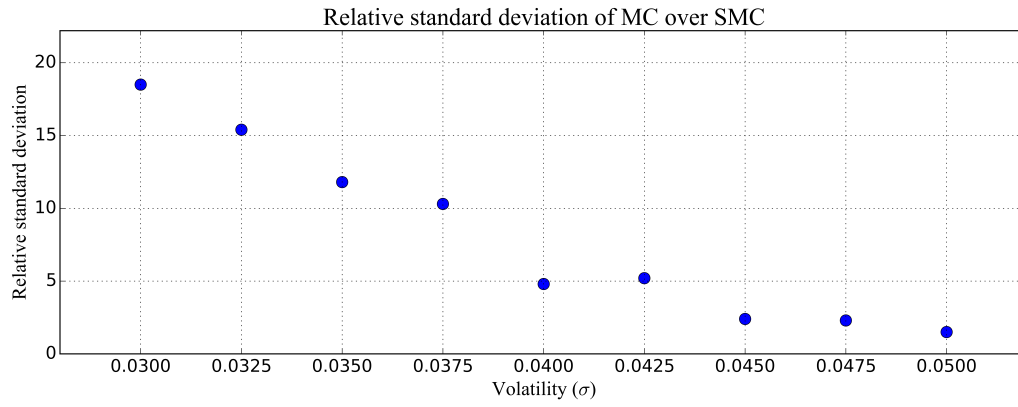


Figure 5: Relative standard deviations of MC over SMC (Algorithm 3) using simple weighting functions for TARN in the constant volatility case.

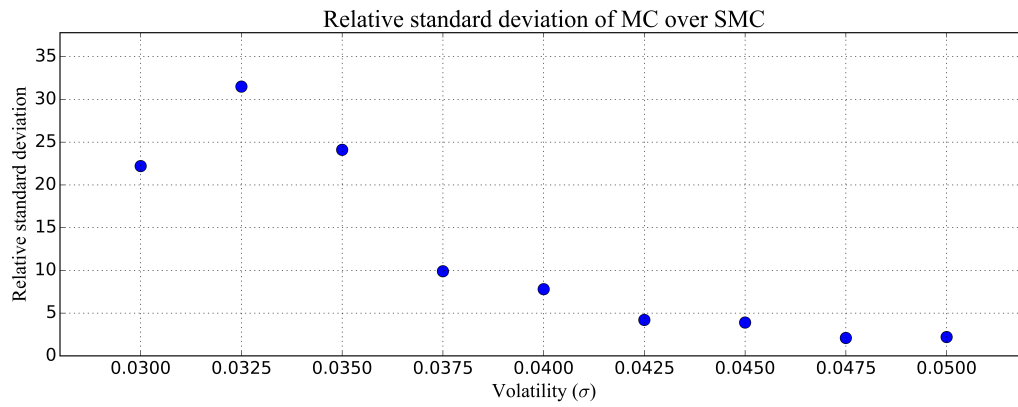


Figure 6: Relative standard deviations of MC over SMC (Algorithm 3) using a mixture of normal densities as the target for TARN in the constant volatility case.

a mixture of normals (in this case, the proportions 0.3 and 0.7) and repeat the experiments. We observe that SMC has a standard deviation which is 2.33 times less than that of MC.

The running times for SMC are twice that of usual MC in this case. This implies that while there are some gains to be achieved in using SMC in this example, they are not as significant as before. This is because of the fact that the local volatility function is low (in the region of around 0.035) only for a few values of the underlying, which suggests that one should actually give even higher weights to force particles to escape. Nevertheless, this does serve to illustrate the potential benefits of using SMC over MC.

6 Conclusion

We have formally presented the idea of weighting functions in SMC. The main idea is to try to estimate the sequence of optimal importance densities using a relatively heuristic technique of weighting functions, giving more weights to potentially favourable particles. We demonstrated this on two examples from finance, but this can be extended to other cases (for instance those in [16, 17]). We have also seen that as we get closer to the optimal importance density the gains keep increasing. However even using an approximation leads to significant gains. Since the idea is quite general, it can potentially be used in finance to price other kinds of path dependent options, for instance Asian options as in [16]. It can also be used in the context of the estimation of marginal expectations w.r.t. laws of jump-diffusions (and hence some control problems); see for instance [17]. The basic notion of weighting functions is also applied in [2] for high-dimensional filtering problems. In summary the ideas in the article provide a simple formalisation of many that have previously appeared in the literature.

There are many extensions of the work in the article. For instance one can adaptively determine the potential functions in the sequence, albeit at the cost of statistical bias. Other ideas include further applications and adaptation to problems where the multi-level Monte Carlo method can be used (e.g. [2]).

A Unbiasedness of Estimate

We provide a proof of the unbiasedness of the estimate when the resampling is done adaptively and multinomially.

A.1 Additional Definitions

Let $\tau_1 < \tau_2 < \dots < \tau_r$ be the successive resampling times and let $\tilde{\tau}(n)$ be the most recent resampling time before time n , with $\tilde{\tau}(n) = 0$ if no resampling has been carried out before time n . Also define $\tau_0 = 0$ and $\tau_{r+1} = N$. Define

$$v_n(x_{1:n}) = \prod_{t=\tilde{\tau}(n)+1}^n \alpha_n(x_{1:n}), \quad V_n^{(m)} = \frac{v_n(X_{1:n}^{(m)})}{M \bar{v}_n},$$

where

$$\bar{v}_n = \frac{1}{M} \sum_{j=1}^M v_n(X_{1:n}^{(j)}).$$

If resampling is carried out at time n , then $\{V_n^{(m)}\}_{m=1}^M$ are the resampling weights.

For any test function $\psi : \mathcal{X}^N \rightarrow \mathbb{R}$, we estimate $\psi_N := E_{\pi_N}[\psi(X_{1:N})]$ by $\hat{\psi}_{OR} = \sum_{m=1}^M V_N^{(m)} \psi(X_{1:N}^{(m)})$ and we estimate the normalizing constant by $Z_N^M = \prod_{s=1}^{r+1} \bar{v}_{\tau_s}$. Define:

$$\tilde{H}_{\tau_s}^{(m)} = \bar{v}_1 \cdots \bar{v}_{\tau_s} h_{\tau_s}(X_{1:\tau_s}^{(m)}), \quad H_{\tau_s}^{(m)} = \bar{v}_1 \cdots \bar{v}_{\tau_s} h_{\tau_s}(\bar{X}_{1:\tau_s}^{(m)}),$$

where

$$h_{\tau_s}(x_{1:\tau_s}) = \frac{1}{\prod_{n=1}^{\tau_s} \alpha_n(x_{1:n})}.$$

Observe that

$$\begin{aligned} \frac{H_{\tau_{s-1}}^{(m)}}{\tilde{H}_{\tau_s}^{(m)}} &= \frac{1}{\bar{v}_{\tau_s}} \times \frac{\prod_{n=1}^{\tau_s} \alpha_n(X_{1:n}^{(m)})}{\prod_{n=1}^{\tau_{s-1}} \alpha_n(\bar{X}_{1:n}^{(m)})} \\ &= \frac{v_{\tau_s}(X_{1:\tau_s}^{(m)})}{\bar{v}_{\tau_s}} \\ &= MV_{\tau_s}^{(m)} \\ \Rightarrow \tilde{H}_{\tau_s}^{(m)} V_{\tau_s}^{(m)} &= \frac{1}{M} H_{\tau_{s-1}}^{(m)}. \end{aligned} \tag{6}$$

Define the likelihood ratio as

$$L_N(x_{1:N}) = \frac{\pi_N(x_{1:N})}{\prod_{n=1}^N q_n(x_n|x_{1:n-1})},$$

and let

$$\begin{aligned} \tilde{\psi}_{OR} &= \frac{1}{M} \sum_{m=1}^M L_N(X_{1:N}^{(m)}) \psi(X_{1:N}^{(m)}) H_{\tau(N)}^{(m)} \\ &= \frac{1}{MZ_N} \sum_{m=1}^M \left[\frac{\gamma_N(X_{1:N}^{(m)})}{\prod_{n=1}^N q_n(X_n^{(m)}|X_{1:n-1}^{(m)})} \psi(X_{1:N}^{(m)}) \frac{\bar{v}_1 \cdots \bar{v}_{\tau_r}}{\prod_{n=1}^{\tau_r} \alpha_n(\bar{X}_{1:n}^{(m)})} \right] \\ &= \frac{\bar{v}_1 \cdots \bar{v}_{\tau_r} \bar{v}_{\tau_{r+1}}}{Z_N} \sum_{m=1}^M \frac{v_N(X_{1:N}^{(m)})}{M \bar{v}_N} \psi(X_{1:N}^{(m)}) \\ &= \frac{Z_N^M}{Z_N} \hat{\psi}_{OR}. \end{aligned}$$

The third equality is because $\tau_{r+1} = N$. Define $A_{\tau_s}^{(m)}$ iteratively as $A_{\tau_0}^{(m)} = 1$ and $A_{\tau_s}^{(m)} = A_{\tau_{s-1}}^{(m)} \left(I_{\tau_{s-1}}^{(m)} \right)$, where we recall that $I_{\tau_{s-1}}^{(m)}$ is the resampled index of the m -th particle at the τ_{s-1} -th resampling step. Let

$$\begin{aligned} \mathcal{F}_{2t-1} &= \sigma \left(\left\{ X_1^{(m)} : 1 \leq m \leq M \right\} \cup \left\{ \left(\bar{X}_{1:\tau_s}^{(m)}, X_{1:\tau_s+1}^{(m)}, A_{\tau_s}^{(m)} \right) : 1 \leq s < t, 1 \leq m \leq M \right\} \right), \\ \mathcal{F}_{2t} &= \sigma \left(\mathcal{F}_{2t-1} \cup \left\{ \left(\bar{X}_{1:\tau_t}^{(m)}, A_{\tau_t}^{(m)} \right) : 1 \leq m \leq M \right\} \right); \end{aligned}$$

these are the σ -fields generated by the random variables associated with the M particles just before and just after the t -th resampling step respectively. Let $\tilde{f}_0(\cdot) \equiv \psi_N$ and define for $1 \leq n \leq N$,

$$\tilde{f}_n(x_{1:n}) = \mathbb{E}_q \left[\psi(\bar{X}_{1:N}) L_N(\bar{X}_{1:N}) | \bar{X}_{1:n} = x_{1:n} \right], \tag{7}$$

where \mathbb{E}_q denotes expectation under the proposal density. Then $\tilde{f}_n(x_{1:n}) = \mathbb{E}_q \left[\tilde{f}_N(\bar{X}_{1:N}) | \bar{X}_{1:n} = x_{1:n} \right]$. Let

$$\begin{aligned} Z_{2s-1}^{(m)} &= \left[\tilde{f}_{\tau_s}(X_{1:\tau_s}^{(m)}) - \tilde{f}_{\tau_{s-1}}(\bar{X}_{1:\tau_{s-1}}^{(m)}) \right] H_{\tau_{s-1}}^{(m)}, \\ Z_{2s}^{(m)} &= \tilde{f}_{\tau_s}(\bar{X}_{1:\tau_s}^{(m)}) H_{\tau_s}^{(m)} - \sum_{j=1}^M V_{\tau_s}^{(j)} \tilde{f}_{\tau_s}(X_{1:\tau_s}^{(j)}) \tilde{H}_{\tau_s}^{(j)}. \end{aligned}$$

A.2 The Main Result

Proposition A.1. $\left\{ \left(Z_k^{(1)}, \dots, Z_k^{(M)} \right), \mathcal{F}_k : 1 \leq k \leq 2r+1 \right\}$ is a martingale difference series and

$$M \left(\tilde{\psi}_{OR} - \psi_N \right) = \sum_{k=1}^{2r+1} \left(Z_k^{(1)} + \dots + Z_k^{(M)} \right)$$

Remark A.1. It follows from the proposition that

$$\begin{aligned} \mathbb{E} \left[\tilde{\psi}_{OR} \right] &= \psi_N \\ \Rightarrow \mathbb{E} \left[\frac{Z_N^M}{Z_N} \tilde{\psi}_{OR} \right] &= \psi_N \\ \Rightarrow \mathbb{E} \left[Z_N^M \tilde{\psi}_{OR} \right] &= Z_N \psi_N. \end{aligned}$$

Taking $\psi \equiv 1$ yields the desired result.

Proof of Proposition A.1. We observe that

$$\begin{aligned} Z_1^{(m)} + \dots + Z_{2r+1}^{(m)} &= \sum_{s=1}^r Z_{2s}^{(m)} + \sum_{s=1}^{r+1} Z_{2s-1}^{(m)} \\ &= \sum_{s=1}^r \left[\tilde{f}_{\tau_s} \left(\overline{X}_{1:\tau_s}^{(m)} \right) H_{\tau_s}^{(m)} - \sum_{j=1}^M V_{\tau_s}^{(j)} \tilde{f}_{\tau_s} \left(X_{1:\tau_s}^{(j)} \right) \tilde{H}_{\tau_s}^{(j)} \right] \\ &\quad + \sum_{s=1}^{r+1} \left[\tilde{f}_{\tau_s} \left(X_{1:\tau_s}^{(m)} \right) - \tilde{f}_{\tau_{s-1}} \left(\overline{X}_{1:\tau_{s-1}}^{(m)} \right) \right] H_{\tau_{s-1}}^{(m)} \\ &= -\tilde{f}_{\tau_0} \left(\overline{X}_{\tau_0}^{(m)} \right) H_{\tau_0}^{(m)} + \sum_{s=1}^{r+1} \tilde{f}_{\tau_s} \left(X_{1:\tau_s}^{(m)} \right) H_{\tau_{s-1}}^{(m)} - \sum_{s=1}^r \sum_{j=1}^M V_{\tau_s}^{(j)} \tilde{f}_{\tau_s} \left(X_{1:\tau_s}^{(j)} \right) \tilde{H}_{\tau_s}^{(j)} \end{aligned}$$

Thus,

$$\begin{aligned} &\sum_{m=1}^M \left(Z_1^{(m)} + \dots + Z_{2r+1}^{(m)} \right) \\ &= -M \tilde{f}_{\tau_0} \left(\overline{X}_{\tau_0}^{(m)} \right) H_{\tau_0}^{(m)} + \sum_{m=1}^M \sum_{s=1}^{r+1} \tilde{f}_{\tau_s} \left(X_{1:\tau_s}^{(m)} \right) H_{\tau_{s-1}}^{(m)} - \sum_{m=1}^M \sum_{s=1}^r \sum_{j=1}^M V_{\tau_s}^{(j)} \tilde{H}_{\tau_s}^{(j)} \tilde{f}_{\tau_s} \left(X_{1:\tau_s}^{(j)} \right) \\ &= -M \tilde{f}_{\tau_0} \left(\overline{X}_{\tau_0}^{(m)} \right) H_{\tau_0}^{(m)} + \sum_{s=1}^{r+1} \sum_{m=1}^M \tilde{f}_{\tau_s} \left(X_{1:\tau_s}^{(m)} \right) H_{\tau_{s-1}}^{(m)} - \sum_{m=1}^M \sum_{s=1}^r \sum_{j=1}^M \frac{1}{M} H_{\tau_{s-1}}^{(j)} \tilde{f}_{\tau_s} \left(X_{1:\tau_s}^{(j)} \right) \\ &= \sum_{m=1}^M \tilde{f}_{\tau_{r+1}} \left(\overline{X}_{\tau_{r+1}}^{(m)} \right) H_{\tau_r}^{(m)} - M \tilde{f}_{\tau_0} \left(\overline{X}_{\tau_0}^{(m)} \right) H_{\tau_0}^{(m)} \\ &= M \left(\tilde{\psi}_{OR} - \psi_N \right) \end{aligned}$$

The second inequality is from 6, and the last equality is because $\tau_0 = 0$ and

$$\begin{aligned} \tilde{f}_{\tau_{r+1}} \left(x_{1:\tau_{r+1}} \right) &= \mathbb{E}_q \left[\psi \left(\overline{X}_{1:N} \right) L_N \left(\overline{X}_{1:N} \right) | \overline{X}_{1:\tau_{r+1}} = x_{1:\tau_{r+1}} \right] = \psi \left(x_{1:\tau_{r+1}} \right) L_N \left(x_{1:\tau_{r+1}} \right) \\ \Rightarrow \tilde{f}_{\tau_{r+1}} \left(X_{1:\tau_{r+1}}^{(m)} \right) H_{\tau_r}^{(m)} &= L_N \left(X_{1:N}^{(m)} \right) \psi \left(X_{1:N}^{(m)} \right) H_{\tilde{\tau}(N)}^{(m)} \quad \text{as } X_{1:\tau_{r+1}}^{(m)} = X_{1:N}^{(m)} \text{ and } \tilde{\tau}(N) = \tau_r. \end{aligned}$$

Let E_M denote expectation under the M particle system. To prove the martingale difference property, we observe that

$$\begin{aligned}
E_M \left[Z_2^{(m)} \middle| \mathcal{F}_1 \right] &= E_M \left[\tilde{f}_{\tau_1} \left(\overline{X}_{1:\tau_1}^{(m)} \right) H_{\tau_1}^{(m)} - \sum_{j=1}^M V_{\tau_1}^{(j)} \tilde{f}_{\tau_1} \left(X_{1:\tau_1}^{(j)} \right) \tilde{H}_{\tau_1}^{(j)} \middle| \mathcal{F}_1 \right] \\
&= E_M \left[\tilde{f}_{\tau_1} \left(\overline{X}_{1:\tau_1}^{(m)} \right) H_{\tau_1}^{(m)} - \frac{1}{M} \sum_{j=1}^M \tilde{f}_{\tau_1} \left(X_{\tau_1}^{(j)} \right) H_{\tau_0}^{(j)} \middle| \mathcal{F}_1 \right] && \text{from (6)} \\
&= E_M \left[\tilde{f}_{\tau_1} \left(\overline{X}_{1:\tau_1}^{(m)} \right) H_{\tau_1}^{(m)} - \frac{1}{M} \sum_{j=1}^M \tilde{f}_{\tau_1} \left(X_{\tau_1}^{(j)} \right) \middle| \mathcal{F}_1 \right] && \text{as } H_0^{(j)} = 1 \\
&= 0.
\end{aligned}$$

The last equality is because the conditional distribution of $\left(\overline{X}_{1:\tau_s}^{(1)}, \dots, \overline{X}_{1:\tau_s}^{(M)} \right)$ given \mathcal{F}_{2s-1} is that of M i.i.d. random vectors which take the value $X_{1:\tau_s}^{(j)}$ with probability $V_{\tau_s}^{(j)}$. Also,

$$\begin{aligned}
E_M \left[Z_3^{(m)} \middle| \mathcal{F}_2 \right] &= E \left[\left\{ \tilde{f}_{\tau_2} \left(X_{1:\tau_2}^{(m)} \right) - \tilde{f}_{\tau_1} \left(\overline{X}_{1:\tau_1}^{(m)} \right) \right\} H_{\tau_1}^{(m)} \middle| \mathcal{F}_2 \right] \\
&= E \left[\left\{ \tilde{f}_{\tau_2} \left(X_{1:\tau_2}^{(m)} \right) - \tilde{f}_{\tau_1} \left(\overline{X}_{1:\tau_1}^{(m)} \right) \right\} \middle| \mathcal{F}_2 \right] H_{\tau_1}^{(m)} \\
&= 0.
\end{aligned}$$

The last equality is because

$$\begin{aligned}
E_M \left[\tilde{f}_{\tau_s} \left(X_{1:\tau_s}^{(m)} \right) \middle| \mathcal{F}_{2(s-1)} \right] &= E_M \left[E_q \left(\psi(X_{1:N}) L_N(X_{1:N}) \middle| X_{1:\tau_s} = X_{1:\tau_s}^{(m)} \right) \middle| \mathcal{F}_{2(s-1)} \right] \\
&= E_q \left(\psi(X_{1:N}) L_N(X_{1:N}) \middle| X_{1:\tau_{s-1}} = X_{1:\tau_{s-1}}^{(m)} \right) \\
&= \tilde{f}_{\tau_{s-1}} \left(\overline{X}_{1:\tau_{s-1}}^{(m)} \right).
\end{aligned}$$

The last equality is by the tower property of conditional expectations. Proceeding in this way, it is seen that $\left\{ \left(Z_k^{(1)}, \dots, Z_k^{(M)} \right), \mathcal{F}_k : 1 \leq k \leq 2r+1 \right\}$ is a martingale difference sequence. \square

References

- [1] BESKOS, A., CRISAN, D. & JASRA, A. (2014). On the stability of sequential Monte Carlo methods in high dimensions. *Ann. Appl. Probab.*, **24**, 1396–1445.
- [2] BESKOS, A., CRISAN, D. JASRA, A., KAMATANI, K. & ZHOU, Y. (2014). Towards a stable particle filter in high dimensions. arXiv preprint.
- [3] BOYLE, P. P. (1977). Options: A Monte Carlo approach. *J. Financ. Econ.*, **4**, 323–338.
- [4] CHAN, H. P. & LAI, T. L. (2013). A general theory of particle filters in hidden Markov models and some applications. *Ann. Statist.*, **41**, 2877–2904.
- [5] DEL MORAL, P. (2004). *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer: New York.

- [6] DEL MORAL, P. (2013). *Mean Field Simulation for Monte Carlo Integration* Chapman & Hall: London.
- [7] DEL MORAL, P., DOUCET, A. & JASRA, A. (2006). Sequential Monte Carlo samplers. *J. R. Statist. Soc. B*, **68**, 411–436.
- [8] DEL MORAL, P., GARNIER, J. (2005). Genealogical particle analysis of rare events. *Ann. Appl. Probab.* **15**, 2496–2534.
- [9] DEL MORAL, P. & SHEVCHENKO, P.V., (2014). Valuation of barrier options using Sequential Monte Carlo. arXiv preprint.
- [10] DEL MORAL, P. & MURRAY, L.M. (2014). Sequential Monte Carlo with highly informative observations. arXiv preprint.
- [11] DOUCET, A. & JOHANSEN, A. (2011). A tutorial on particle filtering and smoothing: fifteen years later. In *Handbook of Nonlinear Filtering* (eds. D. Crisan et B. Rozovsky), Oxford University Press: Oxford.
- [12] DOUC, R. & MOULINES, E. (2008). Limit theorems for weighted samples with applications to sequential Monte Carlo methods. *Ann. Statist.*, **36**, 2344–2376.
- [13] GILES, M. (2008). Multi-level Monte Carlo path simulation. *Operations Research*, **56**(3), 607-617.
- [14] GLASSERMAN, P. (2003). *Monte Carlo Methods in Financial Engineering*. Springer: New York.
- [15] GLASSERMAN, P. & STAUM, J. (2001). Conditioning on one-step survival for barrier option simulations. *Op. Res.*, **49**, 923-937.
- [16] JASRA, A. & DEL MORAL, P. (2011). Sequential Monte Carlo for option pricing. *Stoch. Anal. Appl.*, **29**, 292–316.
- [17] JASRA, A., & DOUCET, A. (2009). Sequential Monte Carlo methods for diffusion processes. *Proc. R. Soc. A*, **465**, 3709–3727.